2

NALUPES – Natural Language Understanding and Processing Expert System

Andrzej Pułka
Institute of electronics, Silesian University of Technology, Gliwice
Poland

1. Introduction

Today, modern electronic devices are supplied with many new sophisticated functions, and expectations of their users are constantly growing. Abilities of natural language handling, i.e. understanding and processing commands given in natural language undoubtedly increase the attraction of such equipment. Moreover, this property makes them useful for those persons who have problems with standard communication, with limited manual dexterity, handicapped or even blind persons. This problem can be extended to other fields of human's activity. Also electronic design automation (EDA vendors work on facilitation of the design process for engineers and simplification of their tools. (Pułka & Kłosowski, 2009) described the idea of such expert system supplied with the speech recognition module, dialog module with speech synthesis elements and inference engine responsible for data processing and language interpreting. That approach is dedicated to system-level electronic design problems. The authors focused on automatic generation of modules based on speech and language processing and on data manipulating.
This chapter focuses on highest level language processing phase - the heart of the system – the intelligent expert system responsible for appropriate interpretation of commands given in natural language and formulation of responses to the user. We concentrate on inference engine that works on the text strings that are far from the lowest, signal level.
Automated processing and understanding of natural language have been recognized for years and we can find these problems in many practical applications (Manning & Schultze, 1999, Jurafsky & Martin, 2000). They belong to hot topics investigated in many academic centers (Gu et al. 2006, Ammicht et al. 2007, Infantino et al. 2007, Neumeier & Thompson 2007, Wang 2007). The main objective of the presented contribution is to develop an expert system that aids the design process and enriches its abilities with speech recognition and speech synthesis properties. The proposed solution is intended to be an optional tool incorporated into the more complex environment working in the background. The goal is to create a system that assists the working.
These objectives can be met with the AI-based expert system consisting of the following components: speech recognition module, speech synthesis module, language processing module with knowledge base (dictionary and semantic rules), knowledge base of the design components and design rules and the intelligent inference engine which ties together entire system, controls the data traffic and checks if the user demands are correctly interpreted.

Source: Expert Systems, Book edited by: Petrică Vizureanu,
ISBN 978-953-307-032-2, pp. 238, January 2010, INTECH, Croatia, downloaded from SCIYO.COM

Next section addresses the entire EDA system architecture and localizes the Natural Language Understanding and Processing Expert System (NALUPES).

2. Entire system architecture

The architecture of entire EDA system presented in (Pułka & Kłosowski, 2009) is depicted in Fig. 1. Main elements belonging to the NALUPES expert system are denoted by shaded area. Certainly, the presented scheme covers all levels of the language processing and synthesis ranging from the signal level through all necessary transformations to phonemes and allophones to the text level (Pułka & Kłosowski, 2009). Because this chapter is devoted to the expert system responsible for data analysis, and not detection of signals and their transformations, we concentrate on text level. The heart of the NALUPES system is the inference engine based on Fuzzy Default Logic (Pułka, 2009). The inference engine works on semantic rules implemented within the system and it cooperates with two additional modules: the speech recognition module and the speech synthesis module. The brief description of levels handled by these modules is given in the next section.

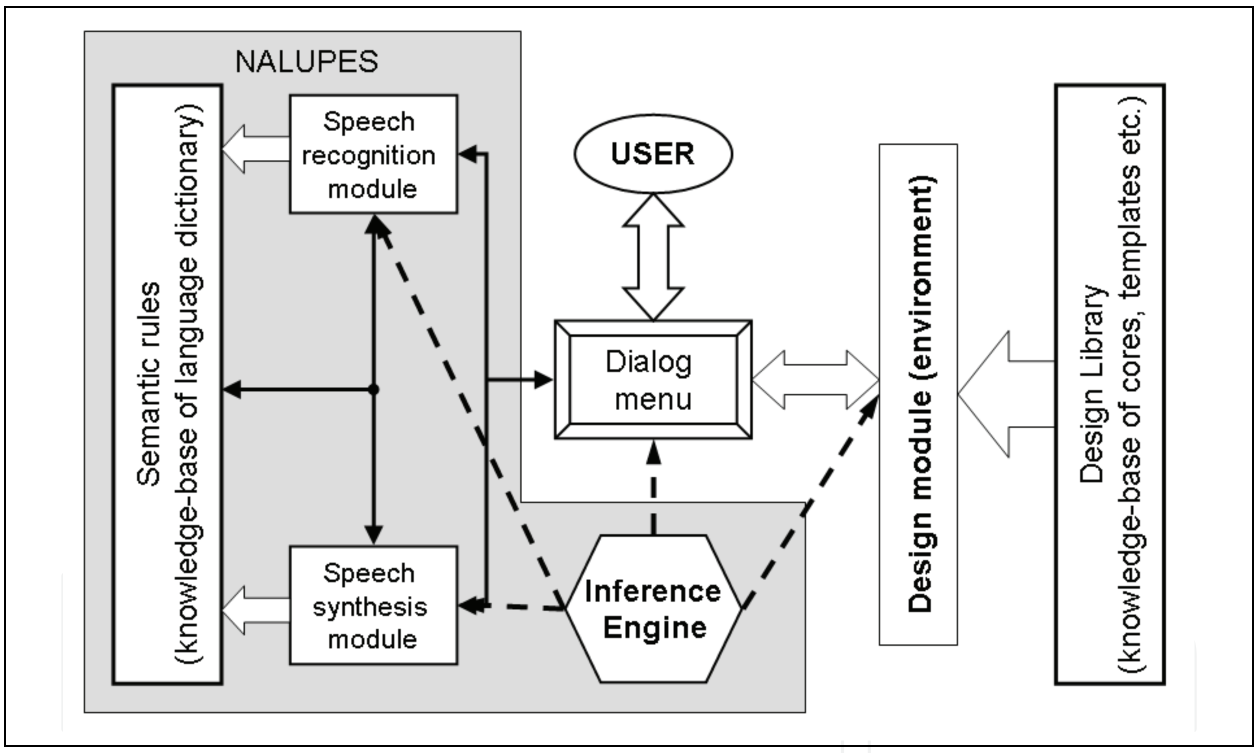
NALUPES
Semantic rules (knowledge-base of language dictionary)
Speech recognition module
Speech synthesis module
USER
Dialog menu
Inference Engine
Design module (environment)
Design Library (knowledge-base of cores, templates etc.)

Fig. 1. A Dialog System Architecture

3. System layers – a brief overview

3.1 Speech recognition module

The speech recognition module is responsible for finding the message information hidden inside the acoustic waveform [4]. The nature of this procedure heavily depends on speaker, speaking conditions and message context. Usually, it is performed in two steps (Fig. 2). In the first step speech signal is transformed into a sequence of phonemes or allophones. Phonemes are sound units that determine meaning of words. In phonetics, an allophone is

one of several similar phones that belong to the same phoneme. A phone is a sound that has a defined wave, while a phoneme is a basic group of sounds that can distinguish words (i.e. change of one phoneme in a word can produce another word). In the second step the sequence of phonemes is converted into the text by phonemes-to-text conversion unit.
The conversion process is of course more complicated and consists of many calculations and involves many modules, among others we have to determine the number of distinctive parameters for each phoneme (Pułka & Kłosowski, 2009), but the obtained text not necessarily is correct, i.e. reflects real meaning and the speaker intentions. In this moment the NALUPES system starts and tries to verify the correctness.

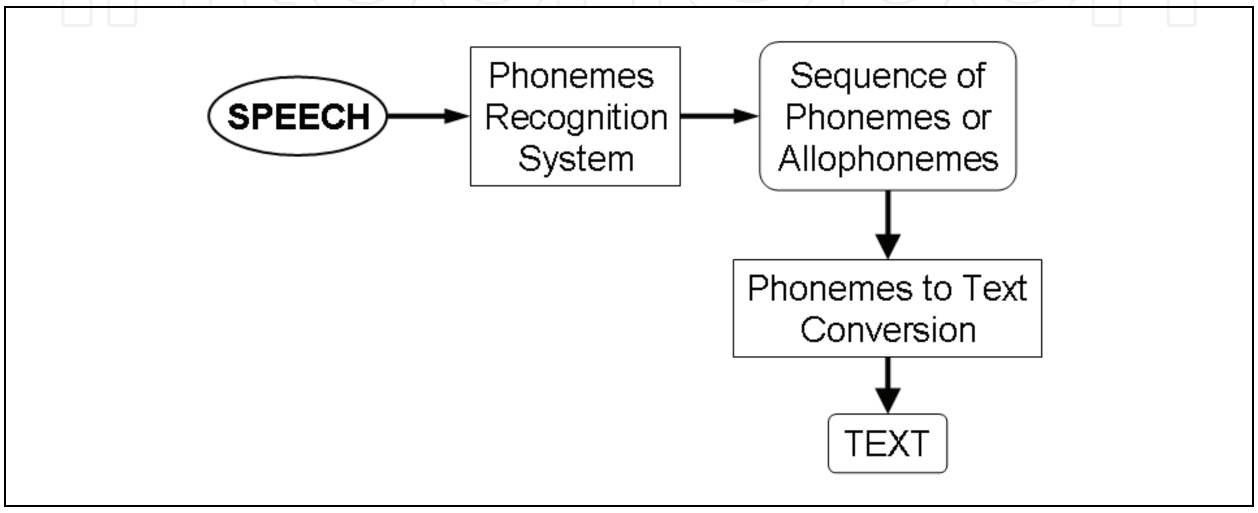
SPEECH
Phonemes Recognition System
Sequence of Phonemes or Allophonemes
Phonemes to Text Conversion
TEXT

Fig. 2. The speech recognition process seen as a transformation speech-to-text (STT).

3.2 Speech synthesis module

The speech synthesis process is widely used in many practical applications, especially in telecommunication devices. Usually the full text-to-speech (TTS) system converts an arbitrary ASCII text to speech. In the first step the phonetic components of the message are extracted and we obtain a string of symbols representing sound-units (phonemes or allophones), boundaries between words, phrases and sentences along with a set of prosody markers (indicating the speed, the intonation etc.). The second step of the process consists of finding the match between the sequence of symbols and appropriate items stored in the phonetic inventory and binding them together to form the acoustic signal for the voice output device (Fig. 3). So, the NALUPES system is responsible for appropriate composition of text strings, and the rest is performed on the phonetic and signal levels, respectively.

4. FDL based inference engine – a heart of the system

As it was mentioned above, the most important part of the NALUPES is its inference engine enriched with sophisticated heuristic tools based on Fuzzy Default Logic (FDL) (Pułka, 2009). The classical logic based approaches usually fail in cases where flexibility is strongly required and the system has to search for a solution which is based on vague and incomplete prerequisite. In our case of phrases recognition, the system is expected to guess the meaning of voice commands even though the original (initial) information is in a useless (incomplete, distorted, mispronounced, misspelled, etc.) form.

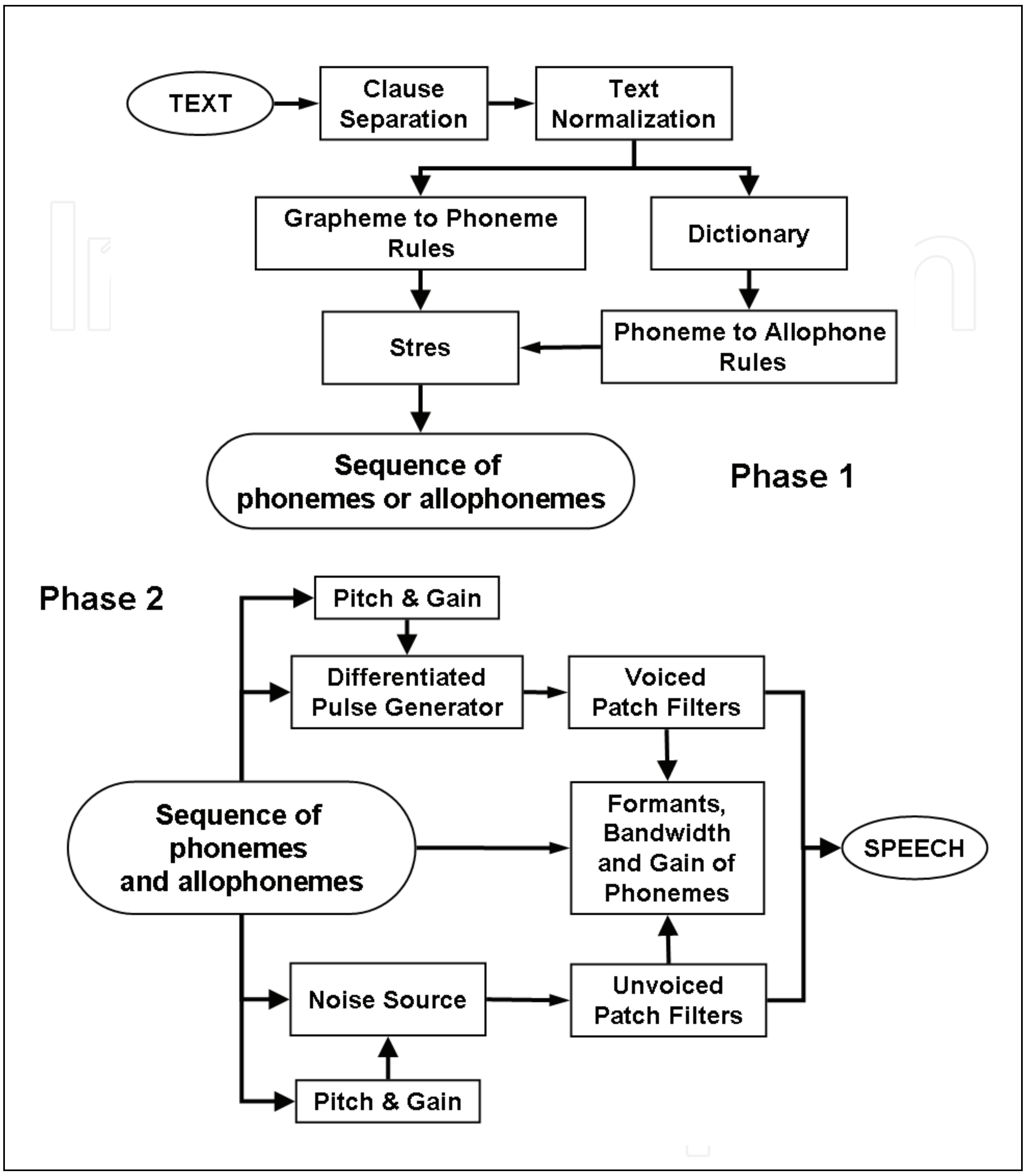
TEXT
Clause Separation
Text Normalization
Grapheme to Phoneme Rules
Dictionary
Stres
Phoneme to Allophone Rules
Sequence of phonemes or allophonemes
Phase 1
Phase 2
Pitch & Gain
Differentiated Pulse Generator
Voiced Patch Filters
Sequence of phonemes and allophonemes
Formants, Bandwidth and Gain of Phonemes
SPEECH
Noise Source
Unvoiced Patch Filters
Pitch & Gain

Fig. 3. The speech synthesis process seen as a two steps transformation text-to-speech (TTS).

4.1 Fuzzy Default Logic (FDL)

Considerations on the one hand on nonmonotonic reasoning and answer set programming, and on the other hand, on fuzzy logic and generalized theory of uncertainty lead to the formulation of Fuzzy Default Logic (Pułka, 2009). This new methodology combines techniques of modeling and handling cases with incomplete information with various types of imprecise information and vagueness. Main definitions of FDL are presented below.

Definition 1

The ***Fuzzy Hypothesis*** (***FH***) is defined as a vector:

$$\Phi^{\lambda} = \left\{ \left[h_1^{\lambda}, \mathrm{Tw}\left(h_1^{\lambda}\right) \right], \left[h_2^{\lambda}, \mathrm{Tw}\left(h_2^{\lambda}\right) \right], \ldots, \left[h_m^{\lambda}, \mathrm{Tw}\left(h_m^{\lambda}\right) \right], \right\} \tag{1}$$

where: $\boldsymbol{h_i^{\lambda}}$ ($\boldsymbol{i}$ = 1...$\boldsymbol{m}$) are wffs in propositional language $\boldsymbol{L}$, and $\boldsymbol{Tw(h_i^{\lambda})}$ denotes ***Trustworthiness***; i.e. one of the modality of generalized constraints in the Zadeh's sense (Zadeh 2006) (bivalent, probabilistic, fuzzy, veristic etc.). For the simplest case the trustworthiness can be treated as a membership function or probability (Zadeh 2008).

Definition 2

The ***Fuzzy Default Rule*** (***FDR***) is the following inference rule:

$$\frac{\alpha : \beta_1, \beta_2 \ldots \beta_{\mathrm{N}}}{\Phi^{\lambda}} \tag{2}$$

$\boldsymbol{\alpha}$, $\boldsymbol{\beta_1 \ldots \beta_N}$ are *wffs* (well formed formulas) in a given propositional language $\boldsymbol{L}$ and $\boldsymbol{\Phi^{\lambda}}$ is a Fuzzy Hypothesis. Moreover, we assume that prerequisite (like in Reiter 2001) is represented by strong information (facts in the sense of (Reiter 1980)), while the possible uncertainty or missing of information is represented by justifications $\boldsymbol{\beta_1 \ldots \beta_N}$. Two assumptions reflect the nonmonotonicity of the inference system: *NaF* (Negation as a Failure) and *CWA* (Closed World Assumption). This scheme reduces the problem of inference path propagation and tracing for trustworthiness. If we would like to have a FDR based fully on ignorance and/or vagueness, the prerequisite is an empty set ($\boldsymbol{\alpha} \equiv \emptyset$).

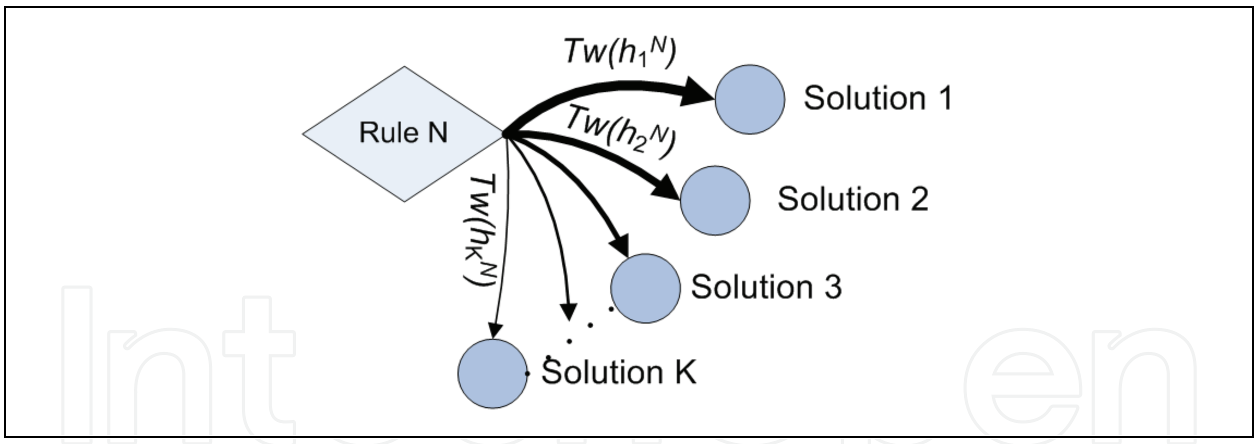

Fig. 4. Hypotheses generation based on FDR scheme – the granular view.

The fuzzy default rules are interpreted similarly to their *classical* predecessors (default rules Reiter 1980). The main difference is in the form of the hypothesis (FH), which consists of different aspects (views, extensions) of the same problem and each of these sub-hypothesis has its own ***Tw*** coefficient. Trustworthiness reflects the *significance* of a given solution, which usually is subjective and can be modified. Elementary hypotheses (components) of a given FH $\boldsymbol{\Phi^{\lambda}}$, i.e. are $\boldsymbol{h_1^{\lambda}}$, $\boldsymbol{h_2^{\lambda}}$,.., $\boldsymbol{h_m^{\lambda}}$ are mutually exclusive. At first glance it looks like inconsistency, because we would like to derive different hypotheses about the same world, but we should remember that each of them has its own trustworthiness level, and moreover, it is the preprocessing phase before the final assessment of hypotheses. In this sense we can call the inference process as a ***multistage procedure***. To eliminate some very weak hypotheses we can add an additional cut-off mechanism (see the implementation) which preserves inferring a hypothesis with a small level of the trustworthiness. Such a solution of the inference engine simplifies the hypothesis derivation and the problem of priorities and existence of various extensions of default theories (Reiter 1980, Brewka 1991) does not limit the application. It is possible, during the inference process, to watch various options (debug the inference engine) and their number can be controlled by the cut-off level. Granular representation of the FDR reasoning procedure presents Fig.4, (the width of arrows corresponds to the values of the trustworthiness).

Definition 3

The ***Credible Set*** (***CS***) $\boldsymbol{H^\lambda}$ of a given Fuzzy Hypothesis $\boldsymbol{\Phi^\lambda}$ is a subset of $\boldsymbol{\Phi^\lambda}$ consisting of those elements $\boldsymbol{h_\iota^\lambda}$ that have appropriate Trustworthiness i.e.:

$$H^\lambda \subseteq \Phi^\lambda \quad \text{and} \quad \underset{h_i^\lambda \in H^\lambda}{\forall} \; Tw\left(h_i^\lambda\right) \geq \text{cut\_off} \tag{3}$$

Naturally, the CS corresponds to those hypotheses that are considered during the further inferring process. The presented mechanisms of hypotheses selection may be more complicated or adjusted dynamically, according to other constraints. The trustworthiness of hypotheses and its ordering corresponds to ordering of literals $\boldsymbol{l_0}$ to $\boldsymbol{l_k}$ in the head of the answer set programming disjunction rules and preference rules (Balduccini & Mellarkod 2004, Balduccini et al. 2006, Gelfond & Lifschitz 1988, Łukasiewicz & Straccia 2008, Van Nieuwenborgh & Vermeir 2006).:

$$\begin{gathered} l_0 \; or \; l_1 \; or \ldots l_k \leftarrow l_{k+1}, \ldots l_m, \; \textbf{not} \; l_{m+1}, \ldots \textbf{not} \; l_n \\ \text{pref}(l_1, l_2);\ \text{pref}(l_2, l_3); \ldots \end{gathered} \tag{4}$$

We will call a given CS ***coherent*** if no its hypothesis is contained in any prerequisite or justification of a FDR, i.e.:

$$\underset{h_i^\lambda \in H^\lambda}{\forall} \Rightarrow \neg \left[\underset{\frac{\alpha^\varphi : \beta_1{}^\varphi, \beta_2{}^\varphi \ldots \beta_N{}^\varphi}{\Phi^\varphi}}{\exists} \left| \begin{matrix} h_i^\lambda \cap \alpha^\varphi \neq \varnothing \\ h_i^\lambda \cap \beta_1{}^\varphi, \beta_2{}^\varphi \ldots \beta_N{}^\varphi \neq \varnothing \end{matrix} \right. \right] \tag{5}$$

Definition 4

The ***hypotheses reduction*** (***HR***) at a given level, we will call the transformation (simplification) of all Credible Sets inferred at this level. This reduction (logical sum of Credible Sets) generates all good (possible for further considerations) hypotheses and reduces the number of trustworthiness per a single hypothesis. After the reduction we obtain all concluded hypotheses that could be considered for the final assessment, and each hypothesis **<u>contains</u>** only one trustworthiness.

$$HR\left\{\bigcup_i H_i^\lambda\right\} = \left\{\left[h_i, Tw(h_i)\right] \middle| \exists H_k^{\lambda_k} \; h_i \in H_k^{\lambda_k} \text{ and } Tw(h_i) = opt\left(Tw\left(h^{\sum_i \lambda_k}\right)\right)\right\} \tag{6}$$

where: $opt\left(Tw\left(h_i^{\sum\lambda_k}\right)\right)$ denotes ***optimal*** value of the trustworthiness for a given element (hypothesis) selected from all Credible Sets.
Selection of optimal function (opt) is another, very interesting problem and it can be a field for user interaction to the inferring process. The optimal function can be flexible, i.e. it can have different meaning for various kinds of trustworthiness (bivalent, probabilistic, veristic, and fuzzy – Zadeh 2008). The optimal function can be also strict (the same for every trustworthiness), which means that it corresponds to one of the following cases: maximum (optimistic approach), mean (no priorities), minimum (worst case analysis), max-min(fuzzy approach) etc.

Definition 5

The ***Fuzzy Default Logic*** (***FDL***) is the theory Δ_{fuzzy} for modeling the commonsense reasoning, which splits the inferring process into stages (steps) Δ^{s}_{fuzzy} and at every stage a given hypothesis is generated. The stage Δ^{s}_{fuzzy} is represented by a quadruple: axioms, simple relations between the knowledgebase elements (classical logic relations), fuzzy default rules and constraints (Apt & Monfroy 2001). The stage Δ^{s}_{fuzzy} is responsible for generation a hypothesis $\boldsymbol{h_s}$. Formally:

$$\Delta_{fuzzy} = \{\, \Delta^{s1}_{fuzzy}, \Delta^{s2}_{fuzzy}, \ldots \Delta^{sN}_{fuzzy}\}; \quad \Delta^{sk}_{fuzzy}\{\, \boldsymbol{A, Facts, FDRs, C} \,\} \mapsto \boldsymbol{h_{sk}} \tag{7}$$

The FDL reminds its ancestors of Default Logic (DL) (Reiter 1980) and Cumulative Default Logic (CDL) (Brewka 1991), however this mechanisms is supplied with vagueness and uncertainty (Zadeh 2004 and Zadeh 2008), and thanks to complex form of hypotheses with their trustworthiness and possibility of generation and consideration of various conclusions it is very close to answer set programming. Instead of preference operator in answer set disjunction logic (Gelfond & Lifschitz 1988), final conclusion is selected on the hypotheses assessment stage. Fig. 5 brings in the scheme of this multi stage inference process. Those pieces of information which are known (stated in a form of axioms, facts or constraints) are not derived. We assume that there is no connections (loop-backs) between FDRs (represented by hexagons) of the same stage, which is very important for the model stability (Pułka 2009). The fuzzy default rules (FDRs) that take part in the inference process (generation of credible sets) are represented by shaded hexagons.

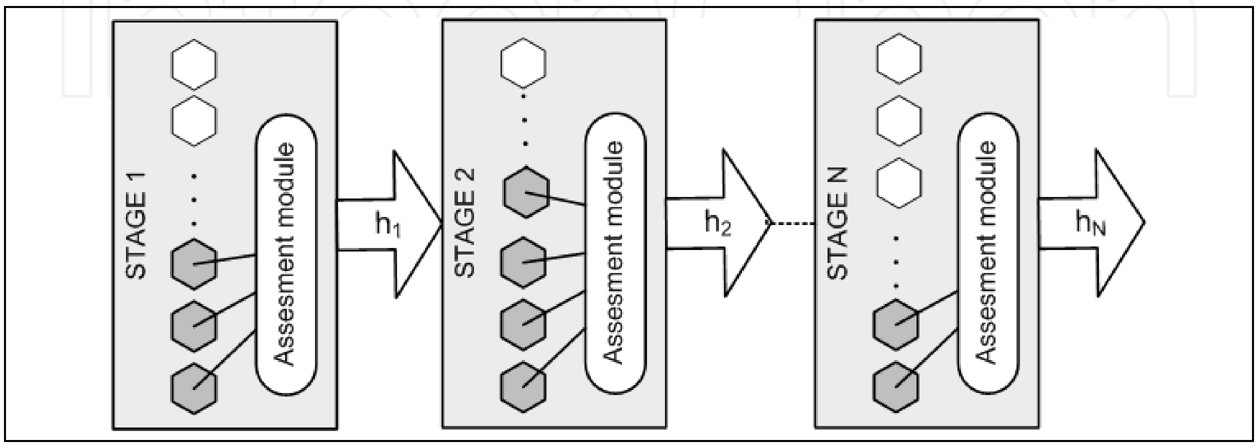

Fig. 5. Generation of Basis Extension (BE) – a multiple stage view.

4.2 Hypotheses generation

As it was mentioned above, very important is the problem of the model stability (Gelfond & Lifschitz 1988), which may lead to unexpected results. The proposed solution (Pułka 2009) is to avoid feedbacks (briefly: concluded hypotheses are not the prerequisites for the other rules) in the inference chain and construction of coherent credible sets. So, the main drawback of the presented methodology is its application specific property, i.e. the implementation of the knowledge base and structure of rules is strongly dependent on the application, and the technique is not a general purpose commonsense modeling. Only the main backbone (shell) of the deduction structure is generalized. On the other hand, however it gives some flexibility to the implementation. In the form presented here, the FDL based inference engine implements the idea of cumulativity (Brewka 1991) and generates so-called Basis Extension (BE) (Pułka & Pawlak 1997). The cumulativity is achieved by calling all nonmonotonic rules and asserting all possible hypotheses as well as justifications (exactly their negations). These *extra* FDL rules control the process of information generation and supplies the system with missing pieces of information. There is no need to keep control on generation rules (priorities of extensions), because every possible hypothesis is considered (the inferring procedure resembles multithread process) and the final selection of the conclusion is performed in the end (assessment phase of Algorithm 1). The implementation requires the appropriate construction of the knowledge-base (addressed in the following section).

Algorithm 1 (Generation of Basis Extension – BE)

```
forall fuzzy_default_rule($\alpha$, $\beta$, $FH^\lambda$)
    if $\alpha$ true then
        if $\beta$ cannot be proved then
            assert every hypothesis $h_\iota^\lambda \in FH^\lambda$ for each
                $Tw(h_\iota^\lambda)$ > cut off level;
update the knowledge-base;
assess inferred hypotheses.
```

4.3 Revision of beliefs

Every formal model of commonsense reasoning has to consider the problem of revision of beliefs, i.e. consistency checking with new pieces of information or occurrence of inconsistency. The meaning of the inconsistency in the fuzzy system should be explained at first. Of course, the existence of two or more mutually exclusive hypotheses is allowed, but we have to remember that after the assessment process, only one conclusion is valid. So, if we find somewhere in the subsequent stages, that the other solution is assumed, we have to reconstruct the deduction chain and verify the hypotheses. The other solution is investigation of more than one paths and parallel analysis of various possibilities. Algorithm 2 given below describes the process of revision of beliefs.

Algorithm 2 (Revision of Beliefs)

```
forall hypothesis ([$h_\iota^\lambda$, $T\varpi$], $Source_i$, $Level_i$) | fuzzy_default_rule($\alpha$, $\beta$, $FH^\lambda$) and $h_\iota^\lambda \in FH^\lambda$
when one of the following conditions is true:
    1° negation of $h_\iota^\lambda$ can be proved or
    2° $\beta$ can be proved or
    3° $\alpha$ cannot be proved
```

remove hypothesis h_ι^λ and every hypothesis hypothesis ([h_κ^λ, $T\omega$], ***Source***$_k$, ***Level***$_k$) such that ***Level***$_k$ > ***Level***$_i$;

update the knowledge-base;
complete (generate) basis extension (BE);
assess inferred hypotheses.

Cases 1°, 2° and 3° represent various situations when consistency of the information has to be verified. This somehow complicated mechanism is forced by the deduction structure within the system that creates a deduction chain. Hypotheses can be generated directly from fuzzy default rules (as their hypotheses) and then we call them as hypotheses of zero level or we can deduce given information basing on the hypothesis which is a result of non-monotonic inference process. The latter has also to be classified as a non-monotonic hypothesis, because is based not on strong fact but on other non-monotonic hypothesis and can be invalidated later. So each hypothesis *has to remember* its predecessor. Because of this deduction structure we can call the inference engine as ***multilevel***. After the revision of beliefs the system is ready to check the completeness of the basis extension and make any necessary supplements, to have a full set of the design information.

4.4 Hypotheses assessment

The generated hypotheses form a deduction chain, and every hypothesis remembers its ancestor. If we denote a hypothesis in a form of a clause hypothesis ([h_ι^λ, $T\omega$], ***Source***$_i$, ***Level***$_i$) it means that the assumed hypothesis has been derived from ***Source***$_i$, which is also a hypothesis of the level ***Level***$_i$. The final assessment and selection of the *best* hypothesis as a final conclusion at a given level can be based on different schemes, which depend on chosen demands: we can take a simple criterion of trustworthiness value (like *verity* in veristic modality of generalized constraints), analyze the entire path (paths) from the ***Source***$_0$ to ***Source***$_i$ and find the global trustworthiness (regarding it like probabilities or possibilities) or use fuzzy criteria max(min) (Łukasiewicz & Straccia 2008, Zadeh 2006). Many examples show that the assessment mechanism gives additional ability to control the model (selected extension). Let's assume that a given FDL model consists of A = ∅; Facts = {C, D, E} and the FDR set contains 3 rules:

(1) (C :B/ {[A_1,0.8], [A_2,0.7], [A_3,0.6]}).
(2) (D :B/ {[A_1,0.4], [A_2,0.6], [A_3,0.4]}).
(3) (E :B/ {[A_1,0.2], [A_2,0.3], [A_3,0.4]}).

So, we have the following trustworthiness for hypotheses A_1, A_2 and A_3, respectively:

$Th(A_1) = \{0.8, 0.4, 0.2\}$.
$Th(A_2) = \{0.7, 0.6, 0.3\}$.
$Th(A_3) = \{0.6, 0.4, 0.4\}$.

The hypotheses reduction process could be controlled by different functions, which are based on altered criteria. This may allow obtaining different solutions as a finally accepted hypothesis: worst case scheme generates A_3 as a positively selected hypothesis with trustworthiness 0.4; the mean criterion based on average value of trustworthiness chooses hypothesis A_2 (average 0.53); and if we employ the classifier based on the best (highest) trustworthiness, we obtain hypothesis A_1 with 0.8 value.

In this sense the assessment of hypothesis gives a lot of flexibility to the user and is application dependant. Moreover, we can split the hypotheses reduction process into two steps: first, selection of the optimal value of the trustworthiness for each hypothesis and then, final assessment and selection of the best one solution. We can use different criteria for each step.

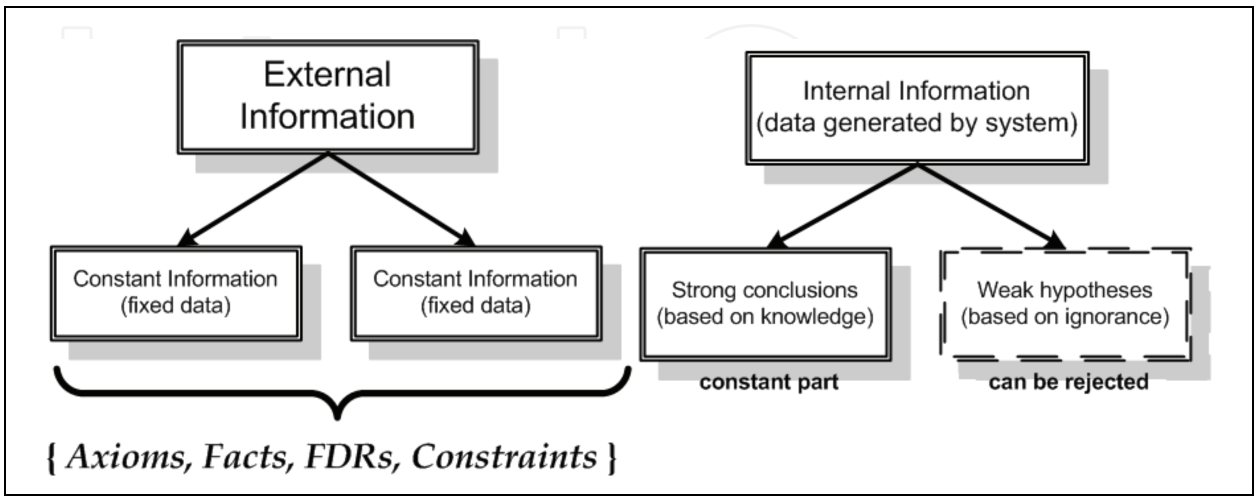
External Information
Constant Information (fixed data)
Constant Information (fixed data)
{ Axioms, Facts, FDRs, Constraints }
Internal Information (data generated by system)
Strong conclusions (based on knowledge)
constant part
Weak hypotheses (based on ignorance)
can be rejected

Fig. 6. The knowledge-base structure.

5. Knowledge-base construction

The knowledge-base and data-base structure on one hand corresponds to the requirements of the inference engine, and on the other hand reflects the properties of the implementation environment (here PROLOG). Details concerning the implementation are addressed in the next section, so let's focus only on main elements constituting the system database.
The entire information stored in the knowledge-base can be divided into two parts: external data acquired by the system from the user and internal data generated by the system. Fig. 6 presents the diagram of the data structure within the entire system. This formal scheme simplifies the information handling and controlling of its consistency (Pułka 2000).

5.1 NLU and NLP rules

The entire dialog system described in Fig.1 consists of many elements handling the voice commands on various levels. The NALUPES system deals only with highest levels of abstraction uses results of modules working on signal level that extract appropriate parameters necessary to recognize and classify phonemes and corrects errors that may occur on lower layers (Fig.7).
The main objectives of the expert system are:
1. correction of unrecognized or wrongly recognized commands;
2. interpretation of entire language phrases constituting commands;
3. understanding and execution of the orders included within commands;
4. preparation of answers or comments to the user (system feedback).

The above goals are achieved during the different phases of the decision process and they relate to the natural language processing (NLP) and the natural language understanding (NLU) processes (Jurafsky & Martin 2008, Bird et al. 2009).

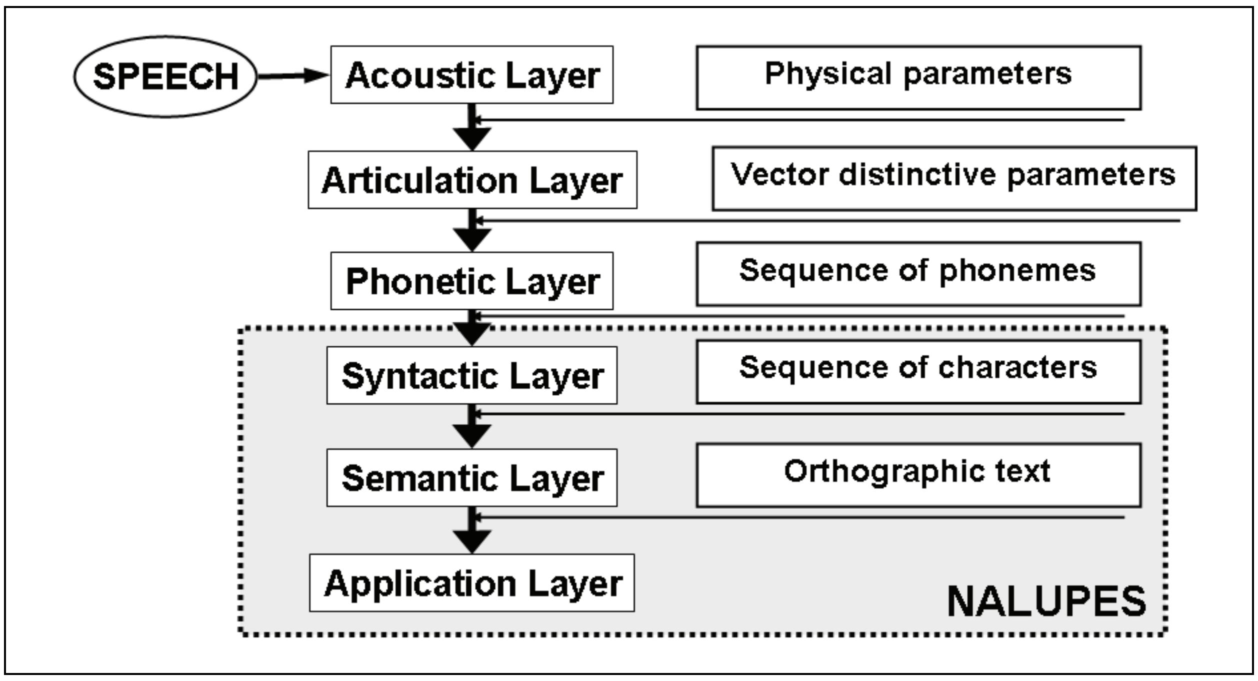

Fig. 7. Multilayer speech recognition process.

First group of tasks concerns recognition of single words. The system should answer the following questions: if a given word exists (belongs to our dictionary)? If a given word has an exact meaning (if a given command makes sense)?
The second phase concerns analyses and verification of sequences of words (entire commands with all necessary and optional parameters). System controls if a given combination of words exists? If this conjunction has a sense?, If it is possible to execute this command within a given phase (context)?
The third group of tasks is responsible for understanding the semantic and meaning of the entire phrase and execution of the appropriate operation. The fourth element of the system generates answers and/or comments to a user or optionally questions in problematic situations.
The system decisions depend on a given context, the menu (here: the design system) and possible actions. Moreover, the deduction structure may look differently for various kinds of situations – there is no unique scheme of the system behavior. However, on the other hand the process of commands correction or understanding is based on templates of phrases and patterns of semantic constructions. The same situation is with the system answers, every response is based on selection of appropriate model from the knowledgebase.
The above scheme justifies the usage of a sophisticated inference mechanism. We can find here: nonmonotonicity (Reiter 1980, Brewka 1991) and incomplete granular information (Kudo & Murai 2004) as well as vagueness and various forms of uncertainty (Zadeh 2006).
Feedback loops present in the verification procedures reflects the nonmonotonic nature of hypotheses (conclusions), which are only temporal and weak. Statistical nature of linguistic information, frequency of appearance of characters, similarity of sounds (allophones and phonemes), the probability of the situation that two or more letters (sounds) are adjacent to each other and many more observations allow incorporating fuzzy models to the processing rules. The scheme depicted in Fig.8 contains small feedback loops (let's say local) and loops covering bigger regions of the deductive process.

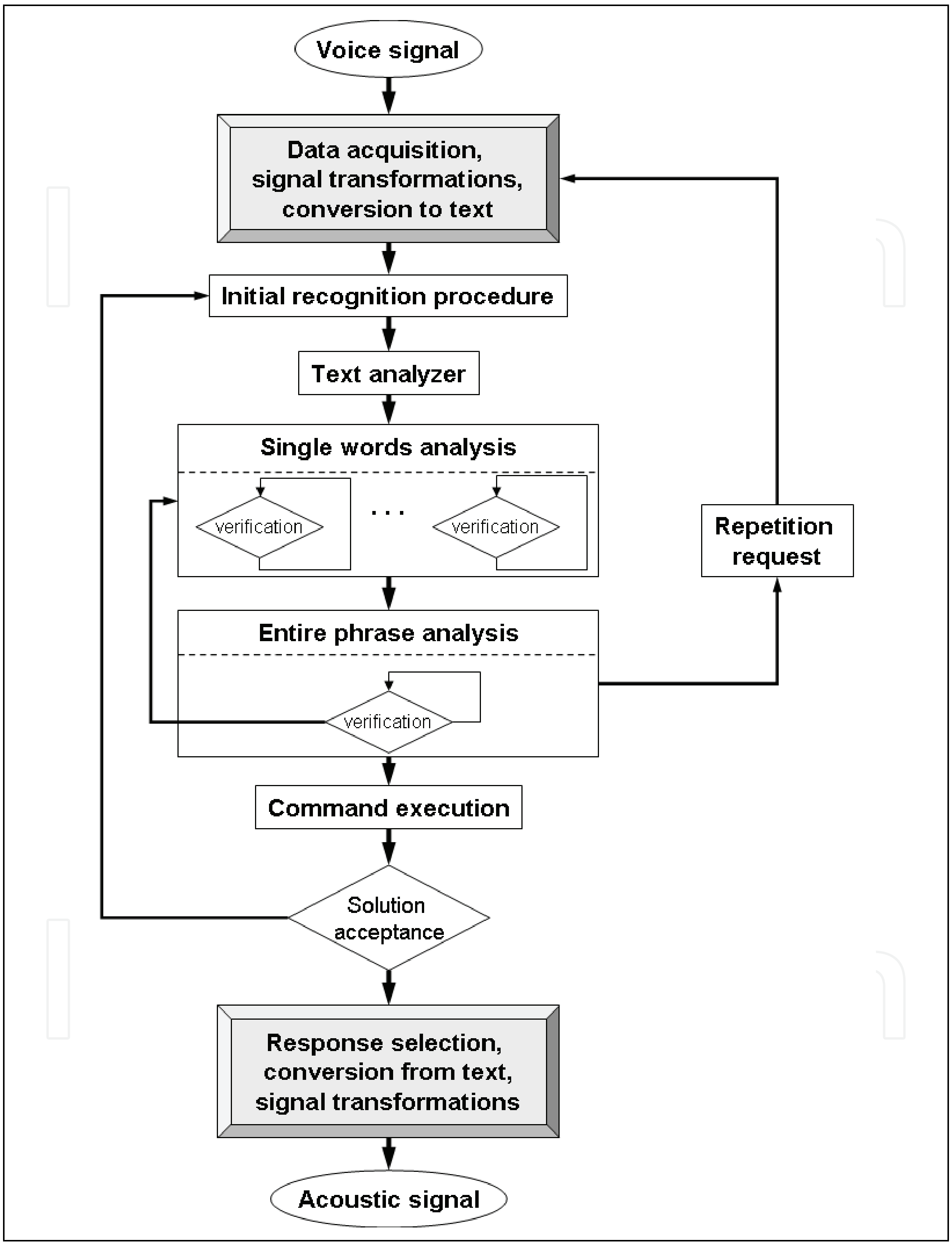

Fig. 8. Natural language processing and understanding sequence.

A single word verification process, which could (but not necessarily must) be executed concurrently, requires examination only short parts of the entire string. In case of any errors, basing on statistical information and current context, the system tries to find in the knowledgebase candidates that best match to a given sequence, i.e. we have an initial chain of characters: $\{C_0, C_1, C_2, \dots C_N\}$ that

- can have a correct meaning,
- can have meaning but with no sense in a given situation,
- can have no meaning (represents no word)

Each of the above cases should be handled and system is expected to select from the templates present in the knowledgebase candidates: $\{\{T^1_0, T^1_1, T^1_2, \dots T^1_N\}, \{T^2_0, T^2_1, T^2_2, \dots T^2_N\}, \dots\{T^k_0, T^k_1, T^k_2, \dots T^k_N\}\}$. These candidates together with their trustworthiness create the fuzzy hypothesis from definition 1. The values of trustworthiness may depend on other factors represented in prerequisite and/or justification, so the FDR rule looks as follows:

$$\frac{\text{Prerequisite} : \text{Justification}}{\Phi^{\lambda}} \tag{8}$$

where: **Prerequisite** represent ***context*** (the menu item, edition phase, evaluation phase or another step of the design process) together with the initial chain $\{C_0, C_1, C_2, \dots C_N\}$; **Justification** could be connected with context (not present elements) or other optional parameters concerning the design process. $\mathbf{\Phi^{\lambda}}$ is a fuzzy hypothesis of the form:
$\{[\{T^1_0, T^1_1, T^1_2, \dots T^1_N\},\mathrm{Tw}_1], [\{T^2_0, T^2_1, T^2_2, \dots T^2_N\},\mathrm{Tw}_2], \dots[\{T^k_0, T^k_1, T^k_2, \dots T^k_N\},\mathrm{Tw}_k]\}$ where each trustworthiness coefficient Tw_i is selected individually depending on a given case (rule). Also the prerequisites could be modeled as a stack with nesting priorities, i.e. ***context***$^{\gamma}$, ***context***$^{\gamma-1}$, ***context***$^{\gamma-2}$, etc. This mechanism allows obtaining another property – controlled deep of backtracking mechanisms.

Process of entire phrase analysis deals with bigger elements: words. Because of the specific nature of the commands to the design system, usually the most important part of a command is the head, i.e. the first element of the chain. The complex semantic tree analysis (Chou & Juang 2002, Gu et al. 2002, Lee & Leong 1992, Bird et al. 2009) is not necessary. We can distinguish three main kinds of commands: actions, descriptions and questions. Furthermore, a command could be simple (trivial case) or complex. In general, the first keyword decides about the command classifying. The analysis is limited to some class of commands, however it is possible to use synonyms (a kind of thesaurus has been implemented). The action commands begin with verbs denoting appropriate activity that should be performed. The description commands denote some execution, but they start with a noun describing a real element in the database of the project. The property of this component has an impact of the analysis of the rest of the chain. The question commands may begin with some limited set of words, like *is*, *what*, *why*, *how* etc. Another commands partitioning presents listing in Fig. 11, we can distinguish global, editing and reviewing commands depending on the current state of the design process.

If the analysis of a given phrase (a chain of words) fails, i.e. gives no real results and corresponds to no order, the verification procedure has to start the revision of beliefs and the system has to invalidate the hypothesis and consider another possibility. This situation may involve deep backtracking and correction of the previous assumptions. The same problem may occur if the solution generated after the execution of the selected command is not satisfying for the user or it contradicts some earlier actions. In such a case, the deduction process could be turned back to previous stages or the user is asked to repeat a command. The latter case usually appears, when several attempt of correction failed.

6. Implementation

The NALUPES system has been implemented in PROLOG language (LPA Prolog) on MS Windows platform. PROLOG language is used for automatic theorem proving, it enables very compact, coherent and consistent implementation of knowledge-base rules. PROLOG is based on linear resolution and uses the backtracking search mechanism. Unfortunately, this mechanism is not very efficient and is not complete – for incompletely defined worlds it can be non-soluble. We can point out some approaches that propose optimization of the search process and reduce the searching space (Prestwich 2001), but the problem of the resolvability is still open in the pure PROLOG. However, in the application domain presented here, where we can assume closed description of the world, this limitation is not very annoying.

6.1 FDL expressed in PROLOG

In order to incorporate the proposed inference mechanisms into the first-order based linear resolution, we have to make few assumptions: the hypothesis of CWA (Closed-world Assumption), existence of more then two logical states for some of the variables representing mutually exclusive solutions and as a consequence the modified logical negation. In other words, the presented fuzzy (cumulative) default logic extends the soundness and strength of the linear resolution and gives new abilities to PROLOG language. The proposed syntax reminds the Cumulative Default Logic engine presented in (Pułka & Pawlak 1997), but introduces the vagueness with Trustworthiness (Fig.9).

```
conclude([Hypothes,Trustworth],Source,Lev):-
        \+(no(Hypothesis)),
        cut_off_level(Cut_Level),Trustworth > Cut_Level,
        \+(hypothesis([[Hypothes,Trustworth],Source,Lev])),
        assertz(hypothesis([[Hypothes,Trustworth],Source,Lev])),
        \+(Hypothesis),
        assertz(Hypothesis).

conclude(_,_,_).

negation(Fact) :- no(Fact),!.          /* modified negation*/
negation(Fact) :-  \+ Fact.
```

Fig. 9. PROLOG implementation of the predicate conclude handling FDR rules.

Fig.9 presents one of the crucial predicates conclude that is responsible for generation of basis extension. It handles fuzzy default rules and a uses the *artificial negation* (predicate no) that complements the gap in PROLOG (PROLOG negation is interpreted as a *negation as a failure*). This philosophy can be used only for limited application and domains that covers *the entire world description*. The more general solution of the negation is still NP-hard problem, however if we assume (and allow) that some parts of knowledge are imprecise and represented not only by two value-logic (true and false), but taking values from more states, we can take advantage of this PROLOG drawback, so the prerequisites as well as conclusions can be of the fuzzy type.

```
phrase_recognize(Phrase):-
            find_list_of_words(Phrase,[Head|Tail),
            menu_select([Head|Tail]).

menu_select([Command|ListOfParameters]):-
            check_command(Command,Menu,MenuType),
            check_sense(Menu,ListOfParameters,Menu1),
            execute(Menu1,ListOfParameters).
/* If the command is not recognized system */
/* generates voice information to the user */
menu_select(_):-
            speech_generate('Command not recognized'),
            speech_generate('Try once more').

check_command(X,X,Type):-                  command(X,Type), !.
check_command(Command,X,_):-  conclude(X, Command).
```

Fig. 10. Examples of PROLOG rules of a semantic checker.

6.2 Knowledge-base rules

The inference engine is responsible for solving *difficult* situations where the system meets problems with recognition and understanding of the user intensions. The significance of a given conclusion depends on the strength of hypotheses that belong to the root of the deduction structure. However the entire process is controlled by linear resolution and all rules are implemented in a form of clauses (Fig. 10). In case of any problems with recognition, the FDL engine is invoked (predicate conclude).

6.3 Database templates

To simplify some recognition operations, the database contains patterns of commands and thesaurus dictionary. The system can find appropriate template and reduce number of fuzzy default engine invocations. The templates has a form of lists (compare to framelists described in (Pułka & Pawlak 1997, Pułka & Kłosowski 2009)). The main problem occurred with names (of modules, ports, signals etc.) that could be often wrongly interpreted, so the system generates subsequent numbers and prescribes fixed names followed by numbers.

```
template(['start', 'edition', 'of', 'data']).
template(['start', 'data', 'edition']).
template(['cancel', 'previous', 'operation']).
template(['insert', 'new', 'port']).
template(['insert', 'new', 'signal']).
template(['add', 'signal', 'name']).
template(['show', 'recently', 'inserted', 'module']).
template(['what', 'is', '10', 'port', 'direction']).
synonym(['start', 'begin', 'initiate', 'set up', 'establish']) .
```

Fig. 11. Examples of the database templates (PROLOG clauses).

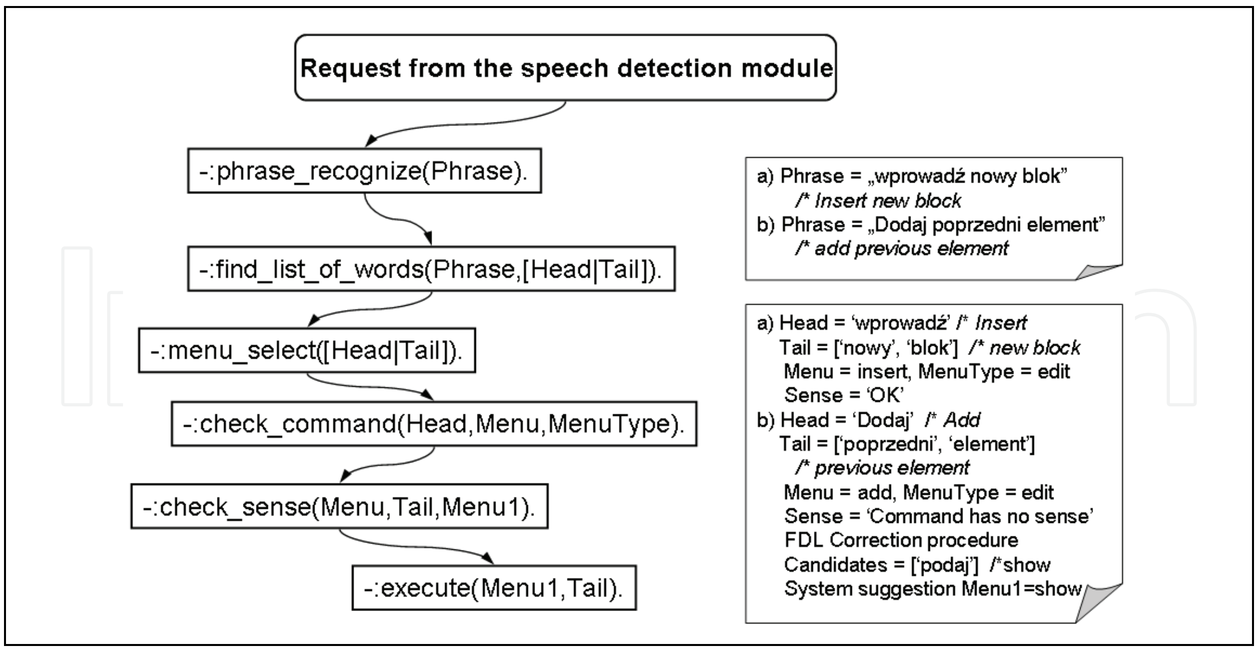
Request from the speech detection module
-:phrase_recognize(Phrase).
-:find_list_of_words(Phrase,[Head|Tail]).
-:menu_select([Head|Tail]).
-:check_command(Head,Menu,MenuType).
-:check_sense(Menu,Tail,Menu1).
-:execute(Menu1,Tail).
a) Phrase = „wprowadź nowy blok” /* Insert new block
b) Phrase = „Dodaj poprzedni element” /* add previous element
a) Head = 'wprowadź' /* Insert
Tail = ['nowy', 'blok'] /* new block
Menu = insert, MenuType = edit
Sense = 'OK'
b) Head = 'Dodaj' /* Add
Tail = ['poprzedni', 'element'] /* previous element
Menu = add, MenuType = edit
Sense = 'Command has no sense'
FDL Correction procedure
Candidates = ['podaj'] /*show
System suggestion Menu1=show

Fig. 12. Examples of the system run.

7. Experiments

Previously, the system has been designed and dedicated to Polish dictionary, it has used characteristic properties of Polish language (Kłosowski 2002). Main advantage of the first version of the system was the very good conversion mechanism at the allophonic level. However the test with modified NALUPES shell (the expert system) supported by FDL inference engine show a radical improvement of the correctly recognized commands. Moreover, the effectiveness could be increased thanks to the user interactions (feedbacks). Currently, the system has been extended to English language and first experiments seem to be very promising.

8. Final remarks

In this chapter the NALUPES – the expert system for understanding and processing of commands in natural language has been presented. The system is based on statistical and numerical semantic language analysis methodologies developed for year and described in literature. However, the main novelty of the system is its sophisticated inference engine which allows improving the efficiency of the recognition process at the textual level.
The area for further investigations is research on system universality, i.e. introduction general rules that allow handling different languages with different databases.

9. References

Ammicht E., Fosler-Lussier E., Potamianos A. (2007). *Information Seeking Spoken Dialogue Systems: Part I and Part II*, IEEE Transactions on Multimedia, Vol.9(3), 2007, pp.532–566.

Apt K. R. and Monfroy E. (2001). Constraint programming viewed as rule-based programming, *Theory and Practice of Logic Programming*, 1(6): pp.713–750, 2001.

Balduccini M. and Mellarkod V. (2004). Cr-prolog2: Cr-prolog with ordered disjunction, In ASP04 Answer Set Programming: *Advances in Theory and Implementation, volume 78 of CEUR Workshop proceedings*, 2004, pp.98–112.

Balduccini M., Gelfond M., and Nogueira M. (2006). Answer Set Based Design of Knowledge Systems, *Annals of Mathematics and Artificial Intelligence*, 2006.

Bird S., Klein E., Loper E. (2009). Natural Language Processing with Python, O'Reilly Media, June 2009.

Brewka G. (1991). Cumulative Default Logic: in defense of nonmonotonic inference rules, *Artificial Intelligence* 50 (1991), pp.183 - 205.

Chou W. and Juang B. H. (2002). *Pattern Recognition in Speech and Language* Processing, Eds. 2002, CRC Press, Inc.

Gelfond, M. and Lifschitz V. (1988). The stable model semantics for logic programming. In Logic Programming, *Proceedings of the Fifth International Conference and Symposium*, R. A. Kowalski and K. A. Bowen, Eds. The MIT Press, Seattle,Washington 1988, pp.1070–1080.

Gu L., Gao Y., Liu F-H. and Picheny M. (2006). *Concept-Based Speech-to-Speech Translation Using Maximum Entropy Models for Statistical Natural Concept Generation*, IEEE Transactions on Audio, Speech and Language Processing, Vol.14, No.2, March 2006.

Infantino I., Rizzo R., Gaglio S. (2007). *A Framework for Sign Language Sentence Recognition by Commonsense Context*, IEEE Transactions on Systems, Man and Cybernetics: Part C: Appl. & Rev., Vol.37, No.5, 2007, pp. 1034–1039.

Jurafsky D. and Martin J. H. (2008). *Speech and Language Processing*, Speech Recognition and Computational Linguistics, Prentice-Hall, 2008.

Kłosowski P. (2002). *Improving of speech recognition process for Polish language*, Transactions on Automatic Control and Computer Science Vol.47 (61), 2002, pp. 111–115.

Kudo Y., Murai T. (2004). Non-monotonic reasoning in prioritized knowledge bases based on granular reasoning, Proc. *of IEEE Intern. Conf. on Fuzzy Systems*, 2004, vol. 1, pp.275–280.

Lee M. C. and Leong H. V. (1992). *NLUS – a Prolog Based Natural Language Understanding System*, Proceedings of 4th IEEE International Conference on Computing and Information, ICCI'92, pp. 204-207.

LPA Prolog. *http://www.lpa.co.uk*.

Łukasiewicz T. and Straccia U. (2008). Tightly Coupled Fuzzy Description Logic Programs under the Answer Set Semantics for the Semantic Web, *International Journal on Semantic Web and Information Systems*, 4(3), 2008, pp. 68–89.

Manning C. D. and Schütze H. (1999). *Foundations of Statistical Natural Language Processing, Cambridge*, MA: MIT Press 1999.

Neumeier K., Thompson C. (2007). *Parameterizing Menu Based Natural Language Interfaces with Location Models*, Proc. of IEEE KIMAS 2007, pp. 236–240.

Prestwich S. (2001). Local Search and Backtracking vs. Non-Systematic Backtracking, *Proceedings of AAAI 2001 Fall Symposium on Using Uncertainty Within Computation*, 2-4 November, AAAI Press 2001, pp. 109-115.

Pułka A., Pawlak A. (1997). Experiences with VITAL Code Generator Controlled by a Nonmonotonic Inference Engine, *Proc. of WORKSHOP on Libraries Component*

Modelling and Quality Assurance with CHDL'97, Toledo, Spain, April 20-25, 1997, pp.309-320.

Pułka A. (2000). *Modeling Assistant - a Flexible VCM Generator in VHDL*, In R. Seepold, N. Martinez (Ed.): Virtual Components Design and Reuse, Kluwer Acad. Pub., 2000.

Pułka A. (2009). Decision Supporting System Based on Fuzzy Default Reasoning, *Proceedings of the HSI'08 Human Systems Interaction Conference*, Catania, Italy, May 21-23, 2009, pp. 32–39.

Pułka A. and Kłosowski P (2009). Polish Speech Processing Expert System Incorporated into the EDA Tool, in: *Human-Computer Systems Interaction Backgrounds and Applications*, Series: *Advances in Intelligent and Soft Computing* , Vol. 60, Hippe, Z. S.; Kulikowski, J. L. (Eds.), *Springer Verlag* 2009, ISBN: 978-3-642-03201-1, pp. 281–293.

Reiter R. (1980). A logic for default reasoning, *Artificial Intelligence* 13, 1980, pp.81–132.

Reiter R. (2001). Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, Cambridge, Massachusetts: *The MIT Press*, 2001.

Van Nieuwenborgh D. and Vermeir D. (2006). Preferred Answer Sets for Ordered Logic Programs*, Theory and Practice of Logic Programming, Volume 6 Issue* 1-2, January 2006, pp. 107–167.

Wang Y. (2007). *A Systematical Natural Language Model by Abstract Algebra*, 2007 IEEE International Conference on Control and Automation, Guangzhou, China, 2007, pp. 1273–1277.

Zadeh L. A. (2004). *Precisiated natural language* (*PNL*), AI Magazine, Vol. 25(3), 74-91, 2004.

Zadeh L. A. (2006). Generalized theory of uncertainty (GTU)--principal concepts and ideas, *Computational Statistics & Data Analysis* 51, 15-46, 2006.

Zadeh, L. A. (2008). Is there a need for fuzzy logic? *Information Sciences: an International Journal*, 178, 13 (Jul. 2008), 2751-2779.

ISBN 978-953-307-032-2
Hard cover, 238 pages
Publisher InTech
Published online 01, January, 2010
Published in print edition January, 2010